FIG. 1

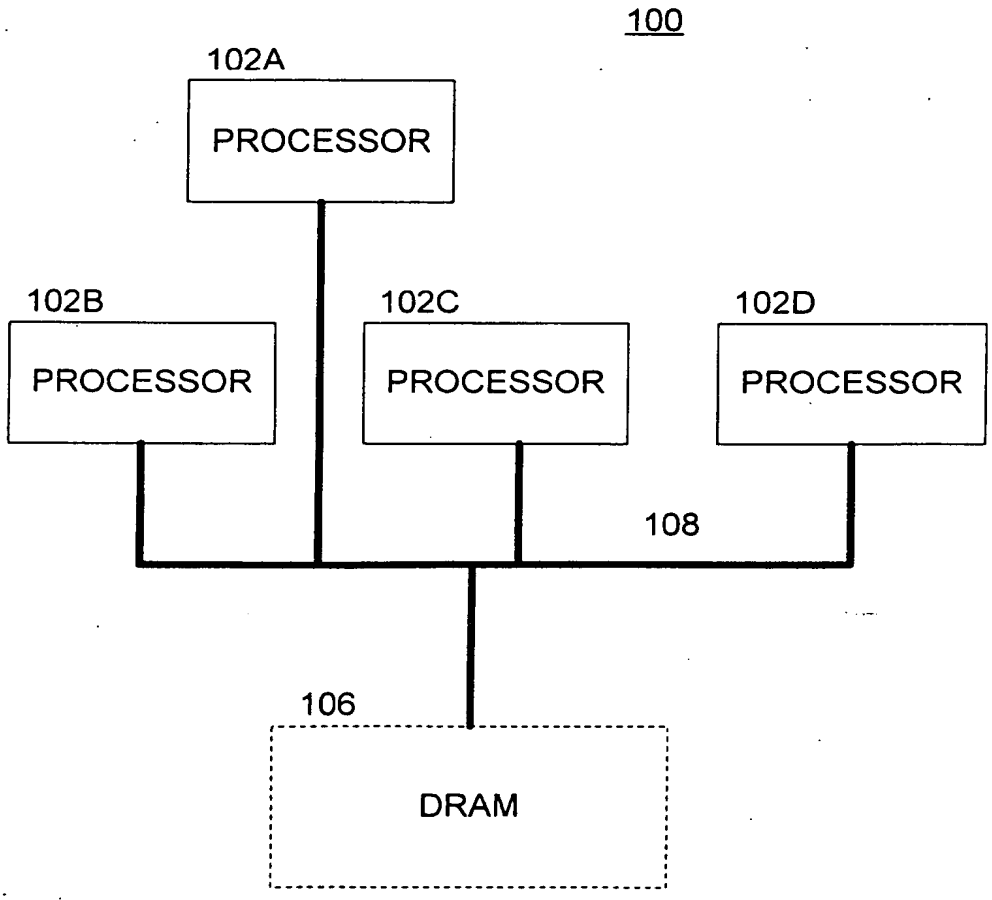

100
102A
PROCESSOR
102B
PROCESSOR
102C
PROCESSOR
102D
PROCESSOR
108
106
DRAM

FIG. 2

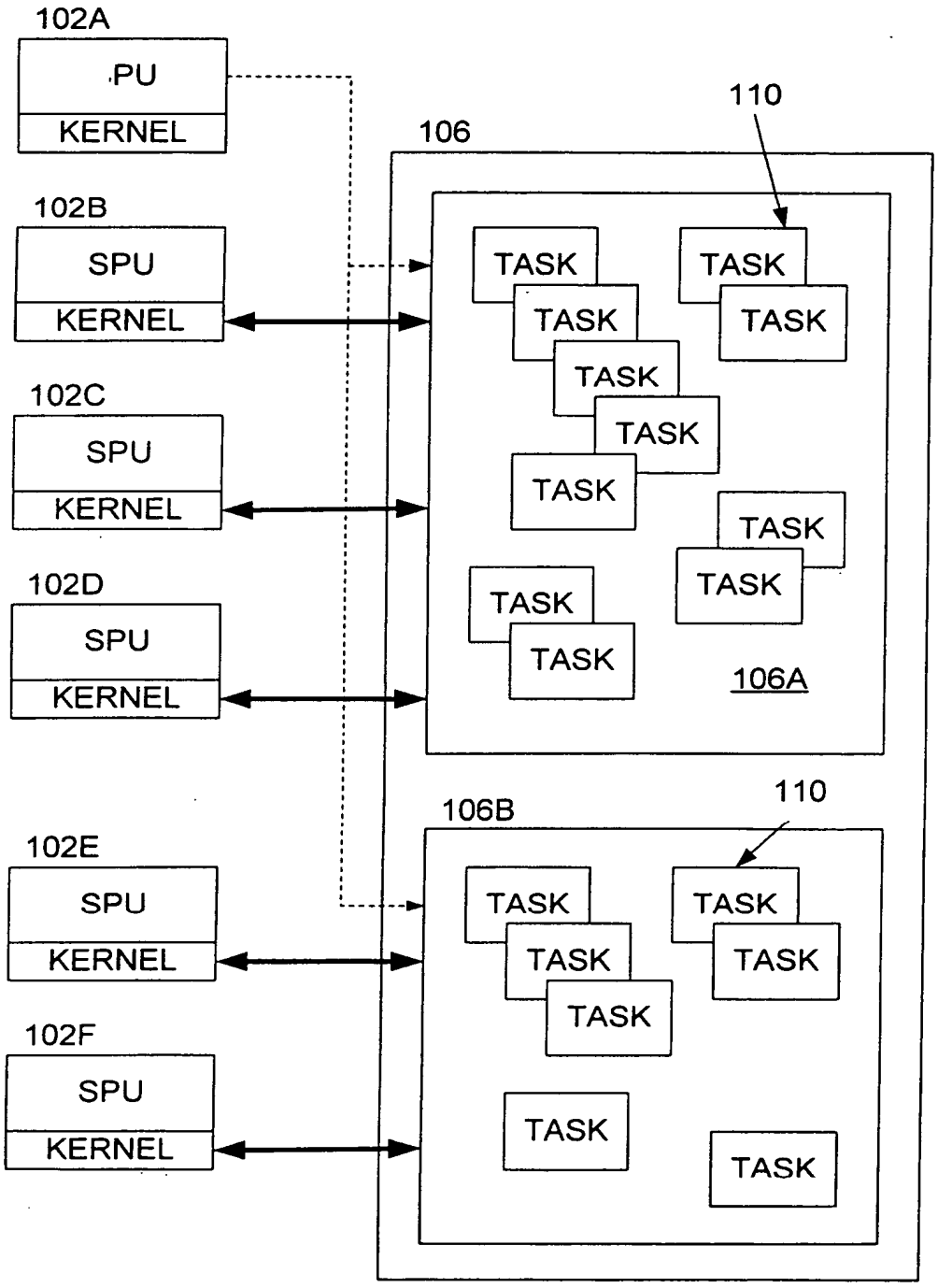

102A
PU
KERNEL
102B
SPU
KERNEL
102C
SPU
KERNEL
102D
SPU
KERNEL
102E
SPU
KERNEL
102F
SPU
KERNEL
106
110
TASK
106A
106B
110
TASK

FIG. 3

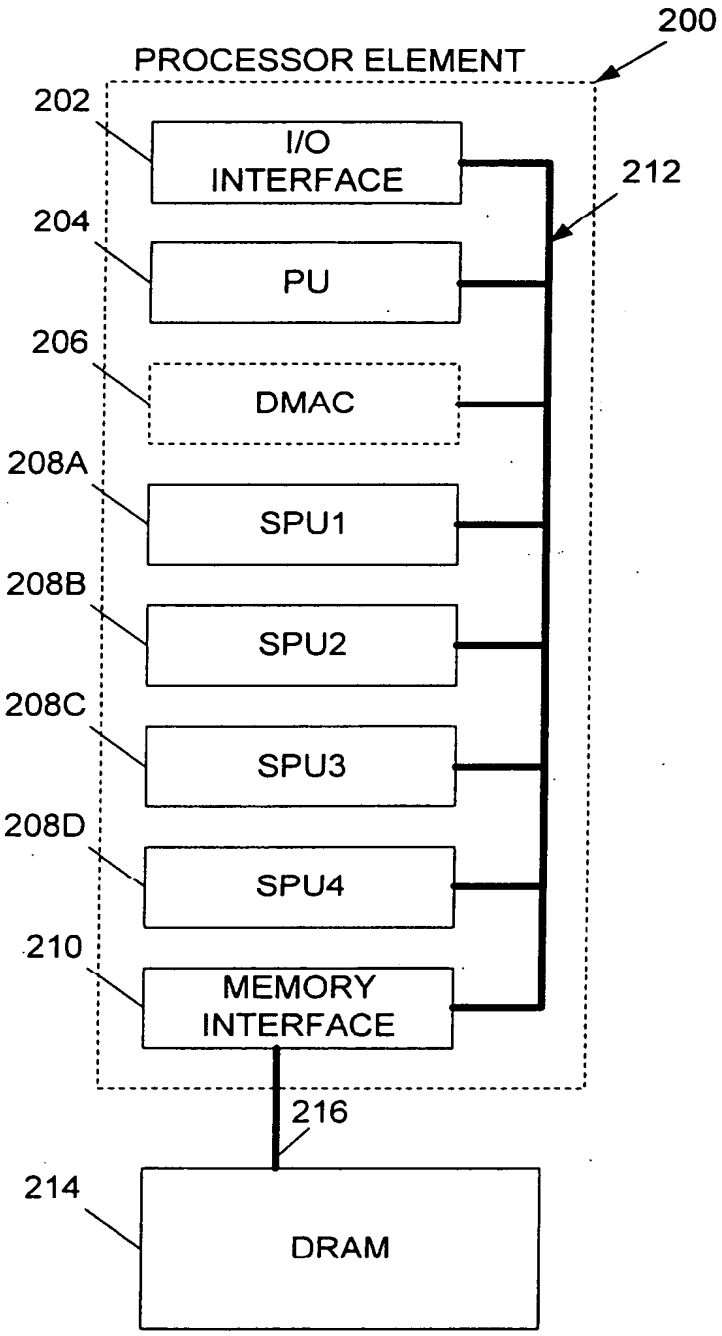

200
PROCESSOR ELEMENT
202
I/O INTERFACE
212
204
PU
206
DMAC
208A
SPU1
208B
SPU2
208C
SPU3
208D
SPU4
210
MEMORY INTERFACE
216
214
DRAM

FIG. 4

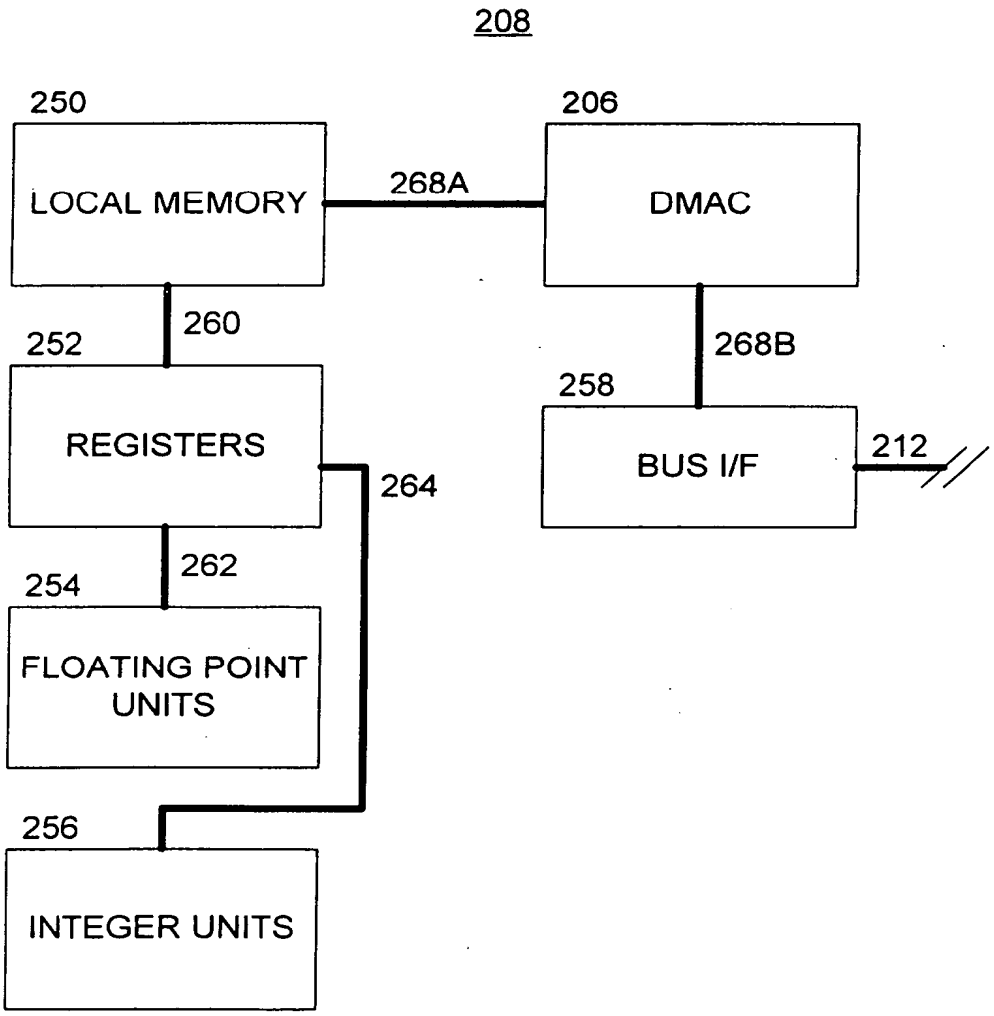

208
250
LOCAL MEMORY
268A
206
DMAC
260
252
REGISTERS
268B
258
BUS I/F
212
264
262
254
FLOATING POINT UNITS
256
INTEGER UNITS

FIG. 5

TASK TABLE

280

T1		T2		T3		T4	
PREV T9	NEXT T8	PREV T4	NEXT T10	PREV T5	NEXT T7	PREV T10	NEXT T2
STAT ready	PRI 1	STAT ready	PRI 2	STAT ready	PRI 3	STAT ready	PRI 2
T5		T6		T7		T8	
PREV T7	NEXT T3	PREV T8	NEXT T9	PREV T3	NEXT T5	PREV T1	NEXT T6
STAT ready	PRI 3	STAT ready	PRI 1	STAT ready	PRI 3	STAT ready	PRI 1
T9		T10					
PREV T6	NEXT T1	PREV T2	NEXT T4				
STAT ready	PRI 1	STAT ready	PRI 2				

FIG. 6

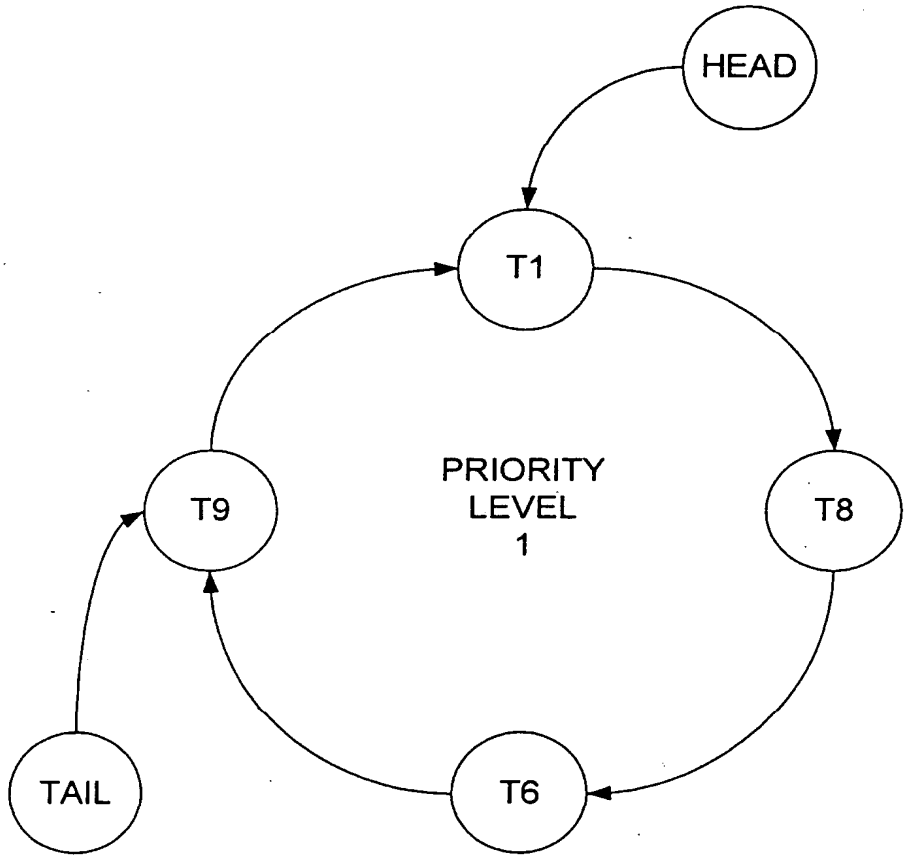


FIG. 7

TASK QUEUE 282

PRIORITY	HEAD	TAIL
1	T1	T9
2	T10	T2
3	T7	T3
4	--	--
5	--	--
6	--	--
7	--	--
8	--	--
. . .		
n	--	--

FIG. 8

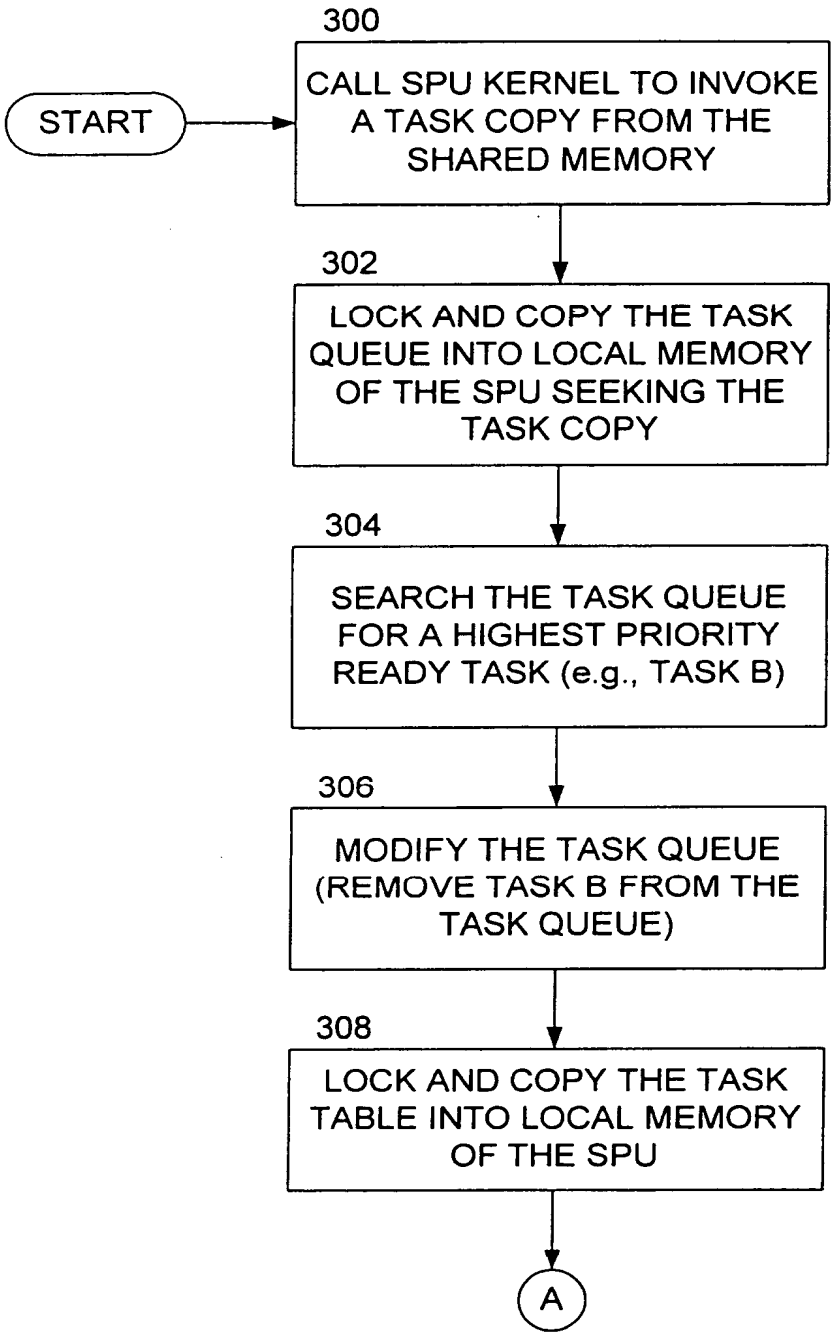

300
START
CALL SPU KERNEL TO INVOKE A TASK COPY FROM THE SHARED MEMORY
302
LOCK AND COPY THE TASK QUEUE INTO LOCAL MEMORY OF THE SPU SEEKING THE TASK COPY
304
SEARCH THE TASK QUEUE FOR A HIGHEST PRIORITY READY TASK (e.g., TASK B)
306
MODIFY THE TASK QUEUE (REMOVE TASK B FROM THE TASK QUEUE)
308
LOCK AND COPY THE TASK TABLE INTO LOCAL MEMORY OF THE SPU
A

FIG. 9

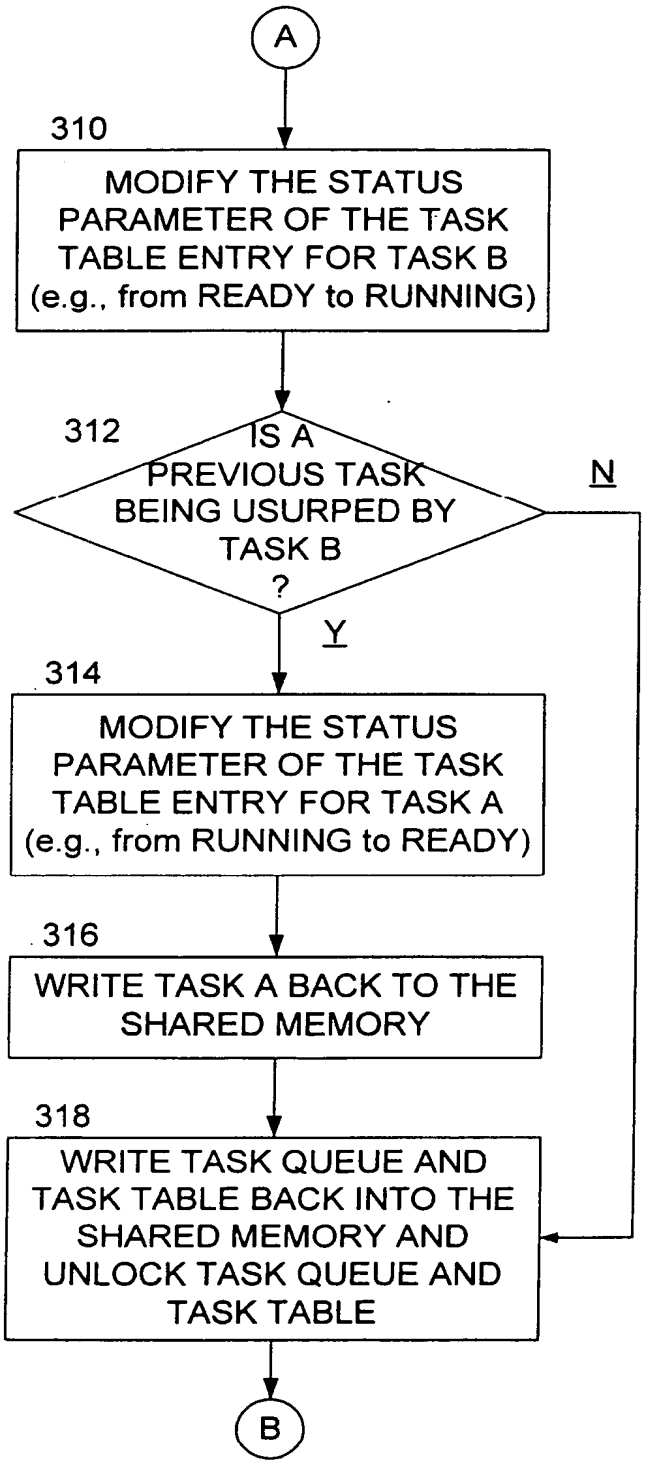

A
310
MODIFY THE STATUS PARAMETER OF THE TASK TABLE ENTRY FOR TASK B (e.g., from READY to RUNNING)
312
IS A PREVIOUS TASK BEING USURPED BY TASK B ?
N
Y
314
MODIFY THE STATUS PARAMETER OF THE TASK TABLE ENTRY FOR TASK A (e.g., from RUNNING to READY)
316
WRITE TASK A BACK TO THE SHARED MEMORY
318
WRITE TASK QUEUE AND TASK TABLE BACK INTO THE SHARED MEMORY AND UNLOCK TASK QUEUE AND TASK TABLE
B

FIG. 10

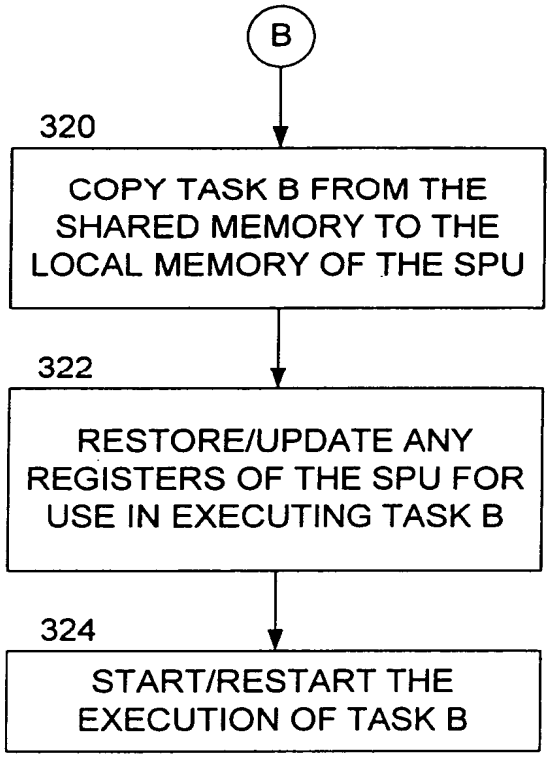

B
320
COPY TASK B FROM THE SHARED MEMORY TO THE LOCAL MEMORY OF THE SPU
322
RESTORE/UPDATE ANY REGISTERS OF THE SPU FOR USE IN EXECUTING TASK B
324
START/RESTART THE EXECUTION OF TASK B

FIG. 11

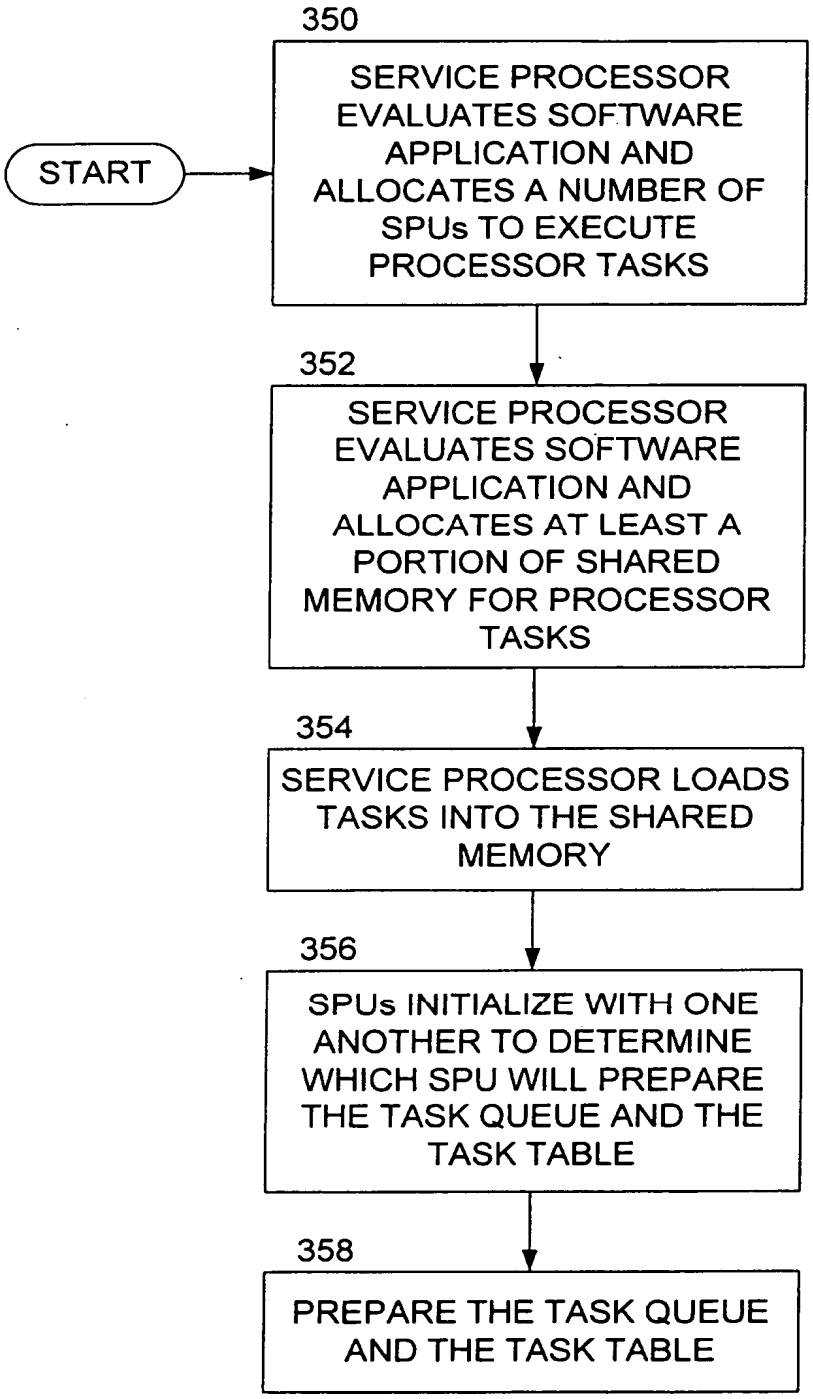

START
350
SERVICE PROCESSOR EVALUATES SOFTWARE APPLICATION AND ALLOCATES A NUMBER OF SPUs TO EXECUTE PROCESSOR TASKS
352
SERVICE PROCESSOR EVALUATES SOFTWARE APPLICATION AND ALLOCATES AT LEAST A PORTION OF SHARED MEMORY FOR PROCESSOR TASKS
354
SERVICE PROCESSOR LOADS TASKS INTO THE SHARED MEMORY
356
SPUs INITIALIZE WITH ONE ANOTHER TO DETERMINE WHICH SPU WILL PREPARE THE TASK QUEUE AND THE TASK TABLE
358
PREPARE THE TASK QUEUE AND THE TASK TABLE

FIG. 12

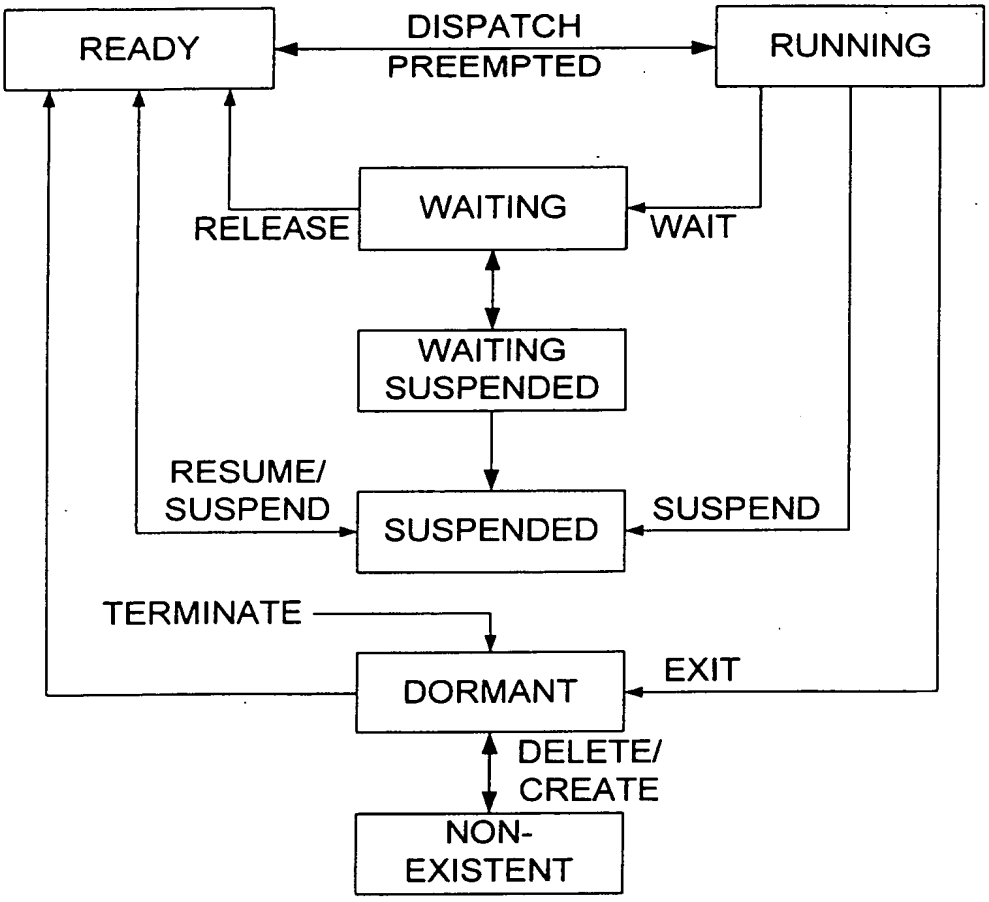

READY
DISPATCH
PREEMPTED
RUNNING
RELEASE
WAITING
WAIT
WAITING
SUSPENDED
RESUME/
SUSPEND
SUSPENDED
SUSPEND
TERMINATE
DORMANT
EXIT
DELETE/
CREATE
NON-
EXISTENT

FIG. 13

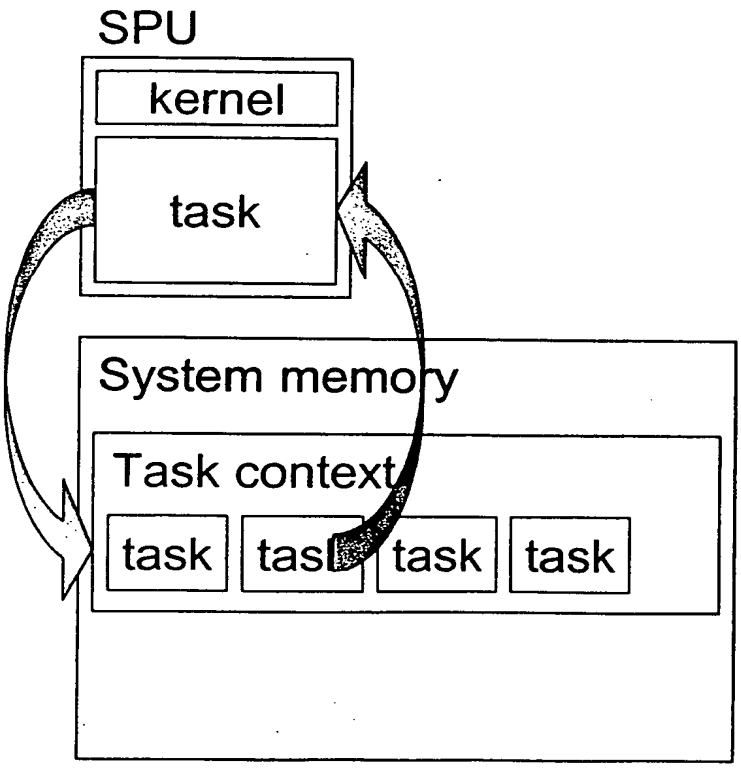

SPU
kernel
task
System memory
Task context
task
task
task
task

FIG. 14

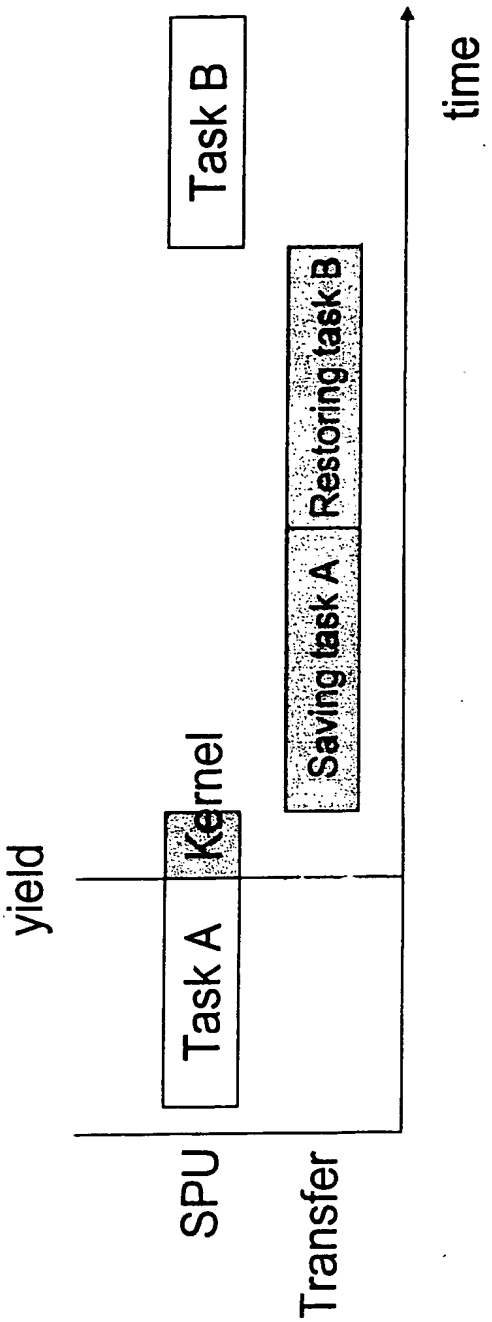

yield
Kernel
Task A
Task B
Saving task A
Restoring task B
SPU
Transfer
time

FIG. 15

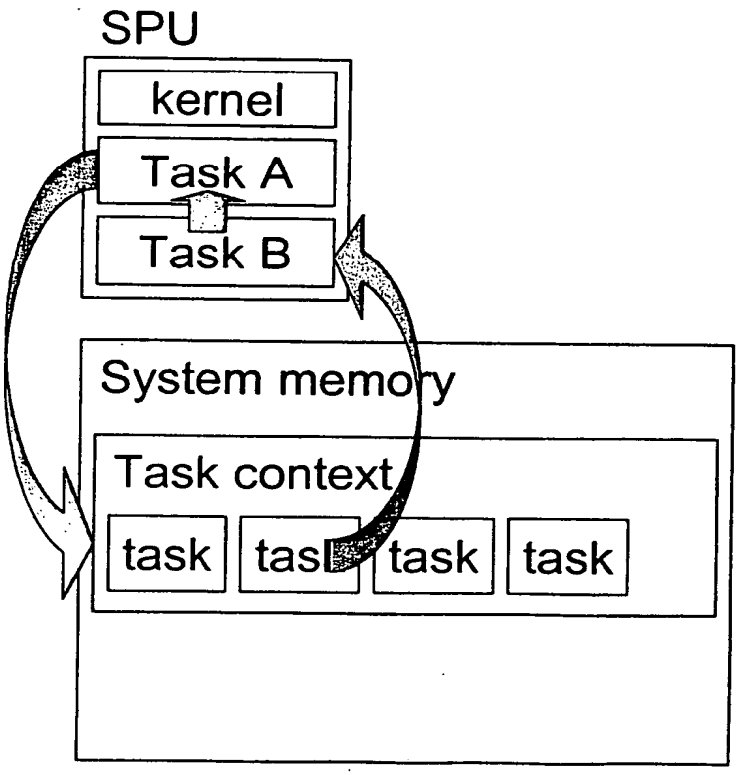

SPU
kernel
Task A
Task B
System memory
Task context
task
task
task
task

FIG. 16

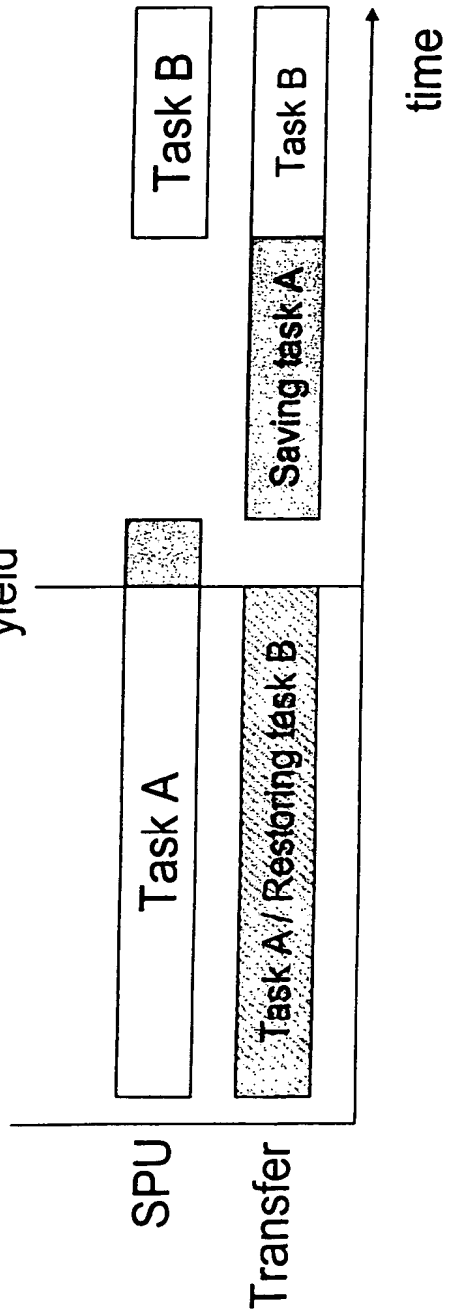

SPU
Transfer
yield
Task A
Task A / Restoring task B
Saving task A
Task B
Task B
time

FIG. 17

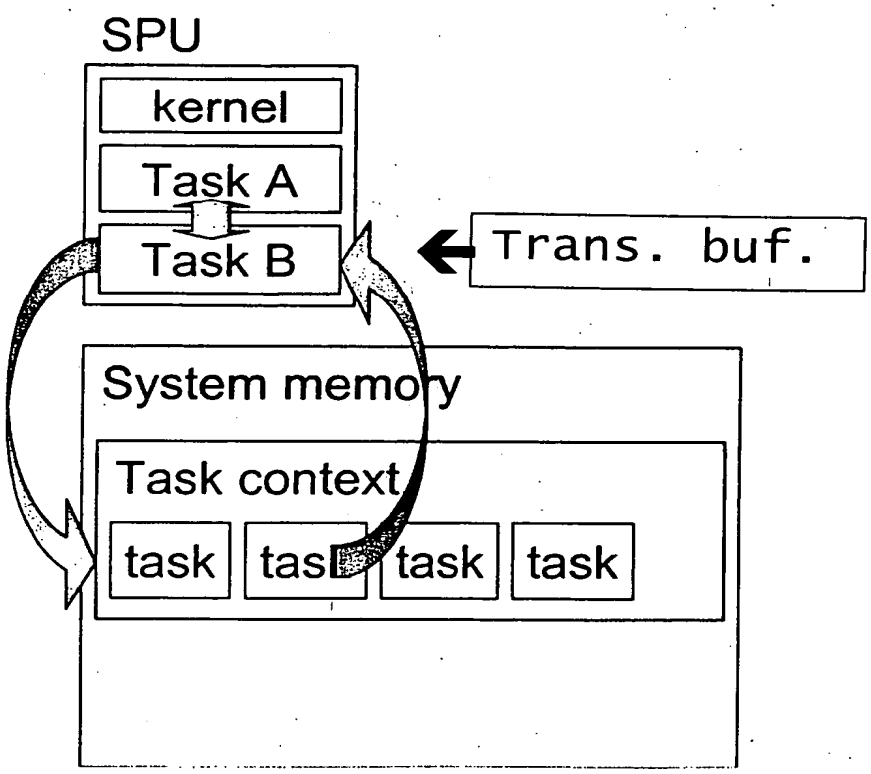

SPU
kernel
Task A
Task B
Trans. buf.
System memory
Task context
task
task
task
task

FIG. 18

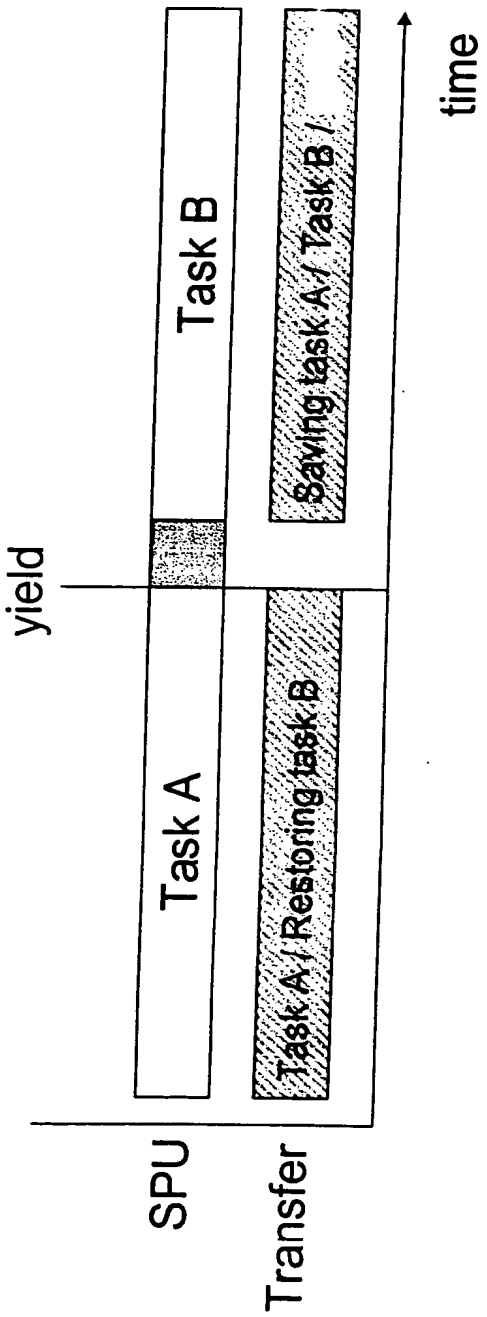

SPU
Transfer
Task A
Task A / Restoring task B
yield
Task B
Saving task A / Task B /
time

FIG. 19

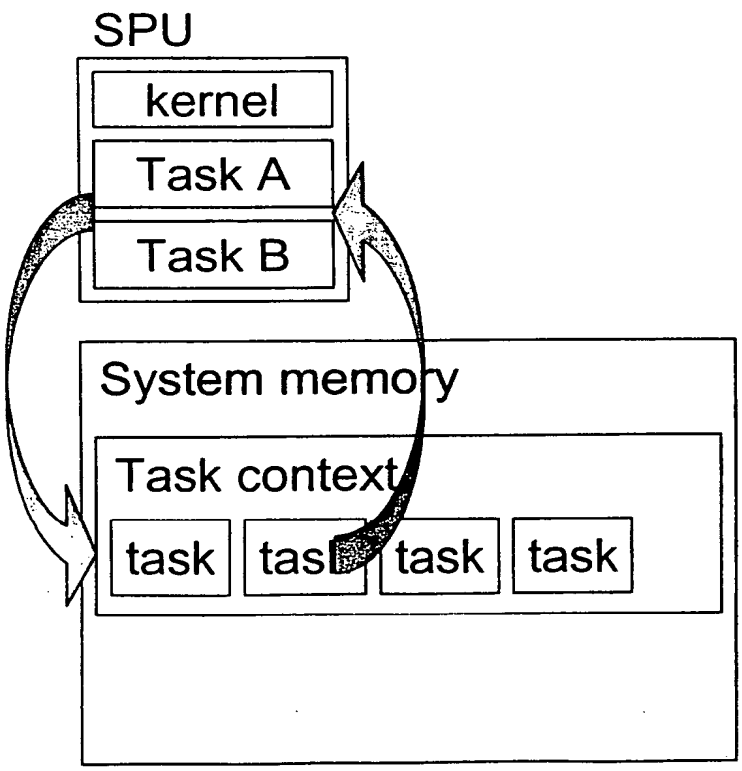

SPU
kernel
Task A
Task B
System memory
Task context
task
task
task
task

FIG. 20

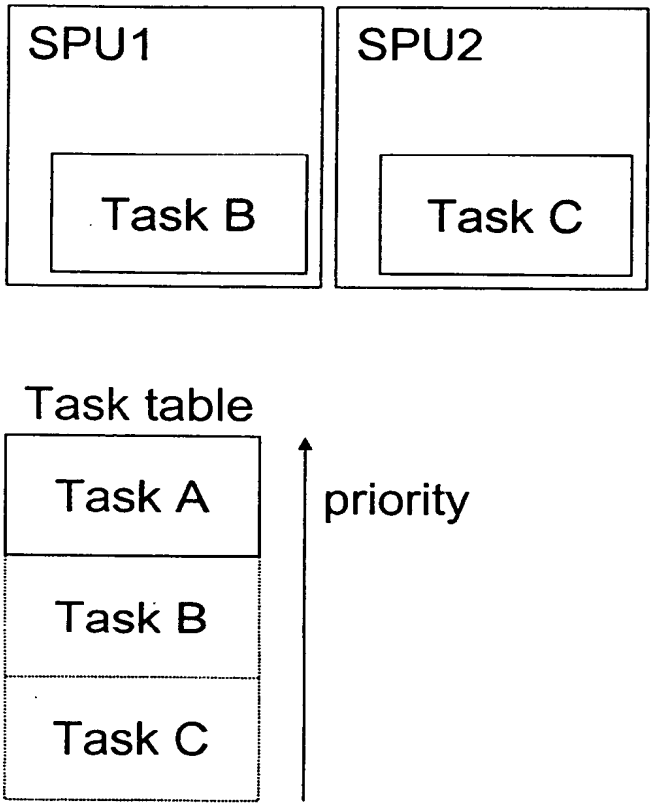

SPU1
SPU2
Task B
Task C
Task table
Task A
Task B
Task C
priority

FIG. 21

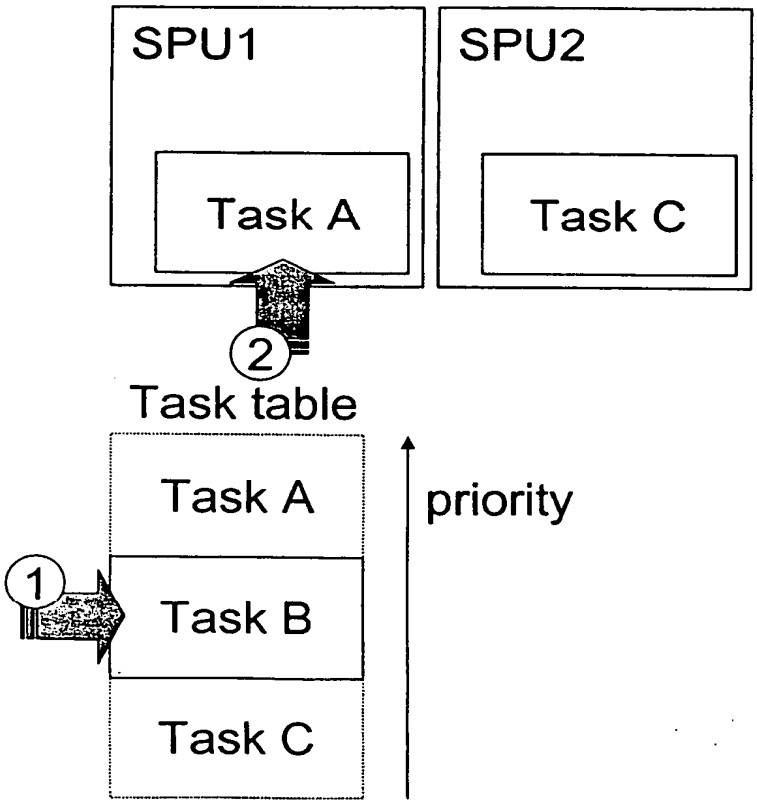

SPU1
SPU2
Task A
Task C
2
Task table
Task A
Task B
Task C
1
priority

FIG. 22

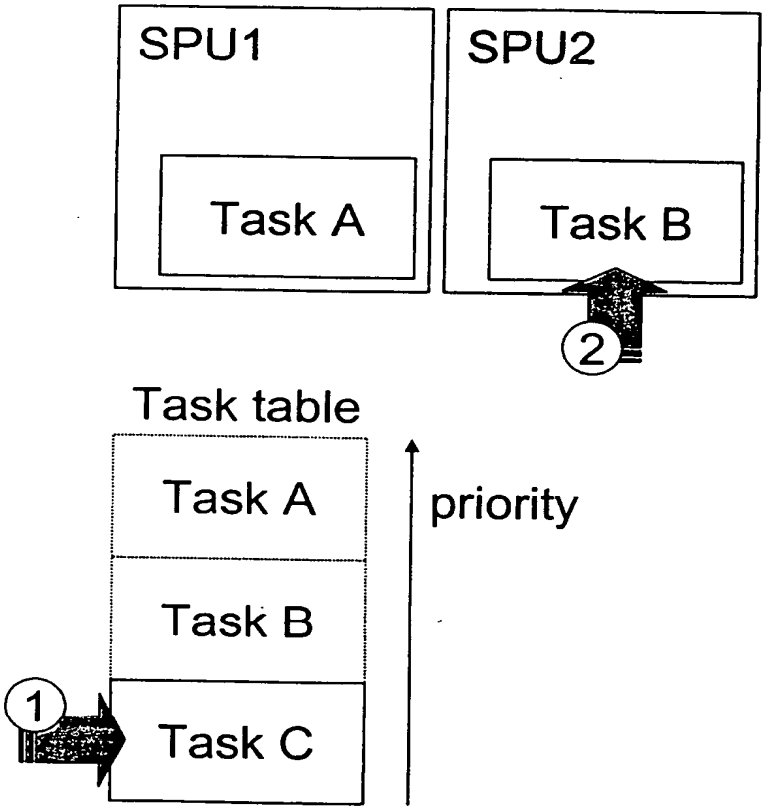

SPU1
SPU2
Task A
Task B
2
Task table
Task A
Task B
Task C
1
priority

FIG. 23

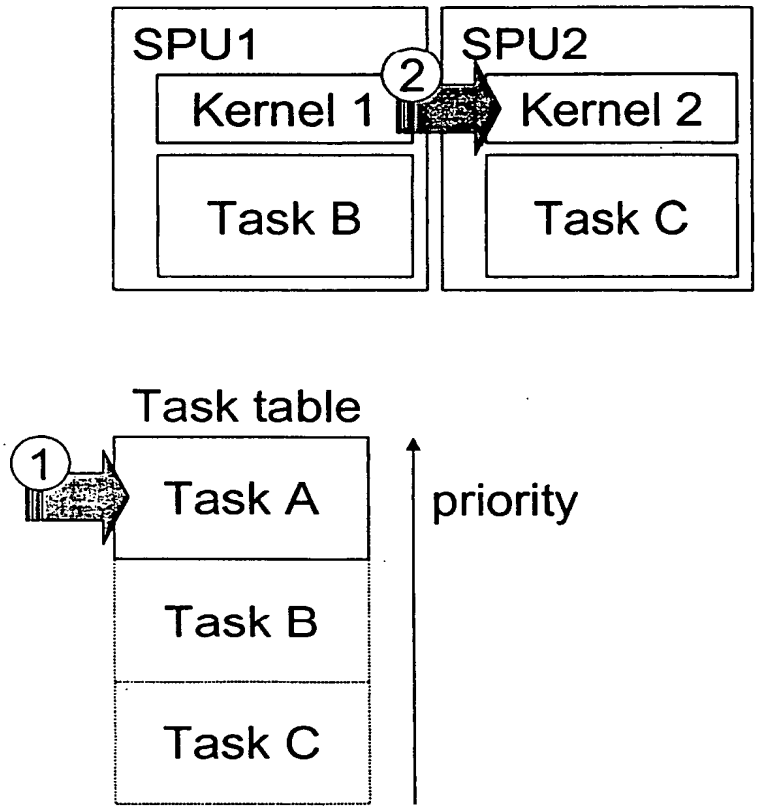

SPU1
SPU2
2
Kernel 1
Kernel 2
Task B
Task C
Task table
1
Task A
priority
Task B
Task C

FIG. 24

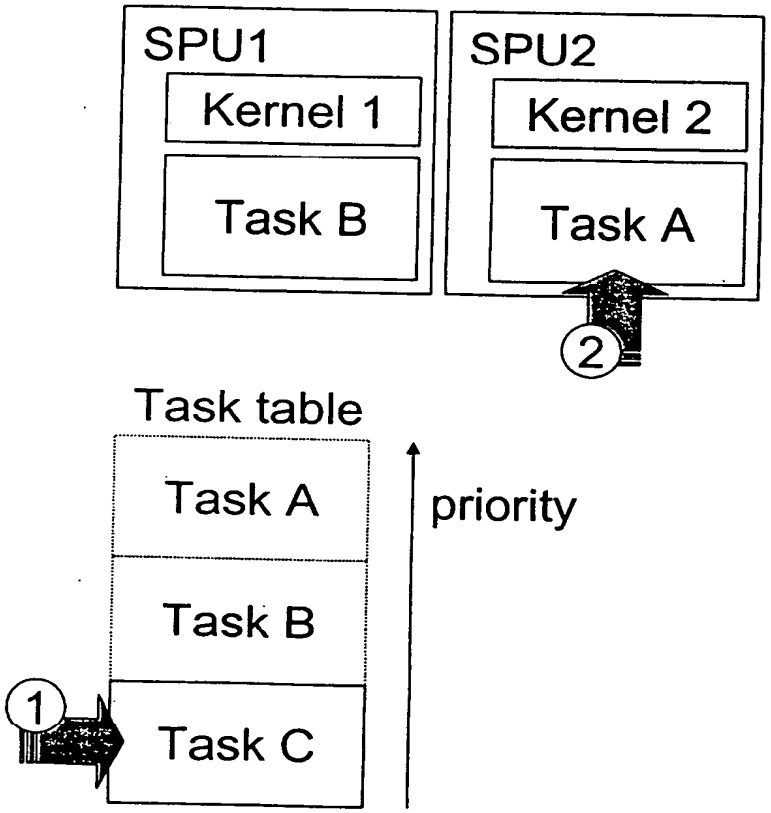

SPU1
Kernel 1
Task B
SPU2
Kernel 2
Task A
2
Task table
Task A
Task B
Task C
1
priority

FIG. 25

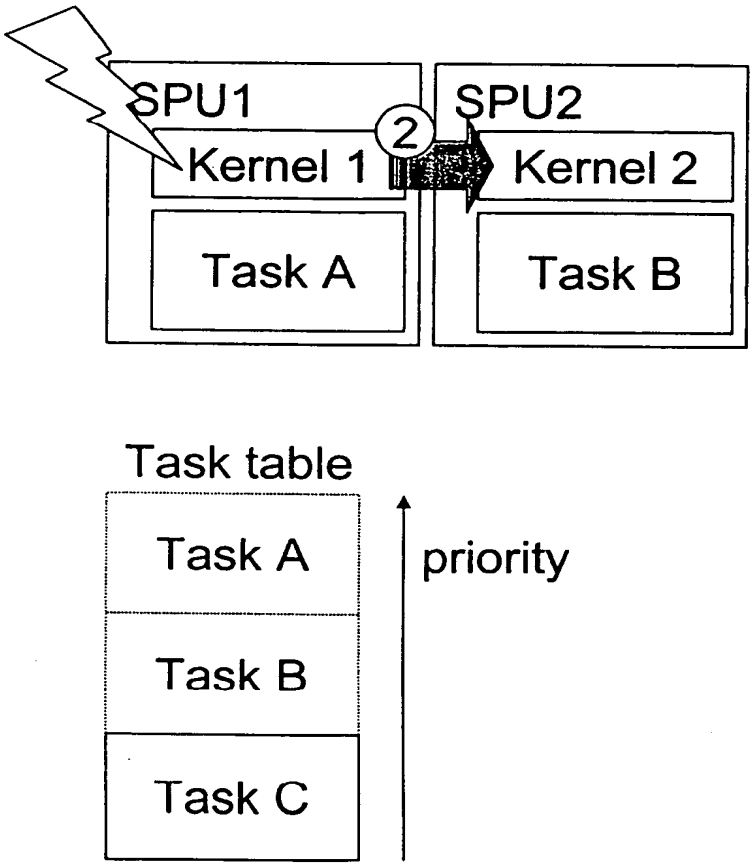

SPU1
SPU2
2
Kernel 1
Kernel 2
Task A
Task B
Task table
Task A
Task B
Task C
priority

FIG. 26

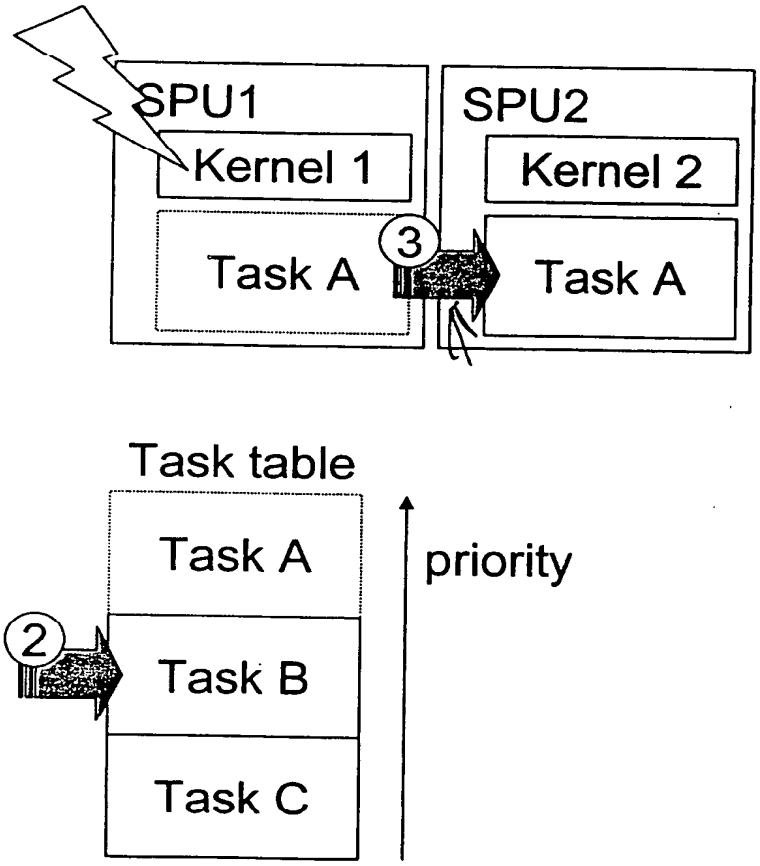

SPU1
Kernel 1
Task A
SPU2
Kernel 2
Task A
3
Task table
Task A
Task B
Task C
2
priority

FIG. 27

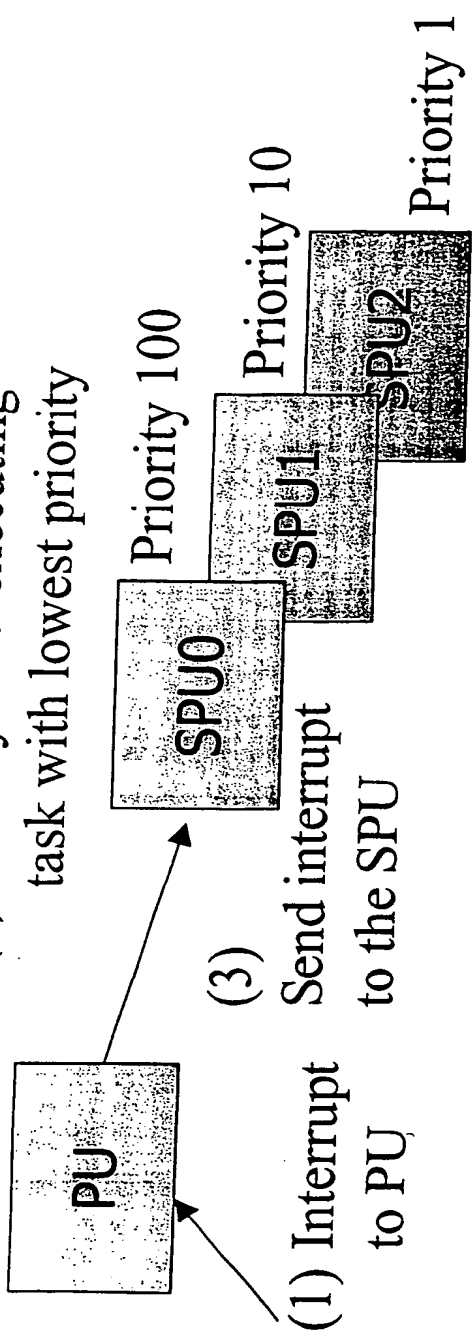

PU
(1) Interrupt to PU
(2) Identify SPU executing task with lowest priority
(3) Send interrupt to the SPU
SPU0
Priority 100
SPU1
Priority 10
SPU2
Priority 1

FIG. 28

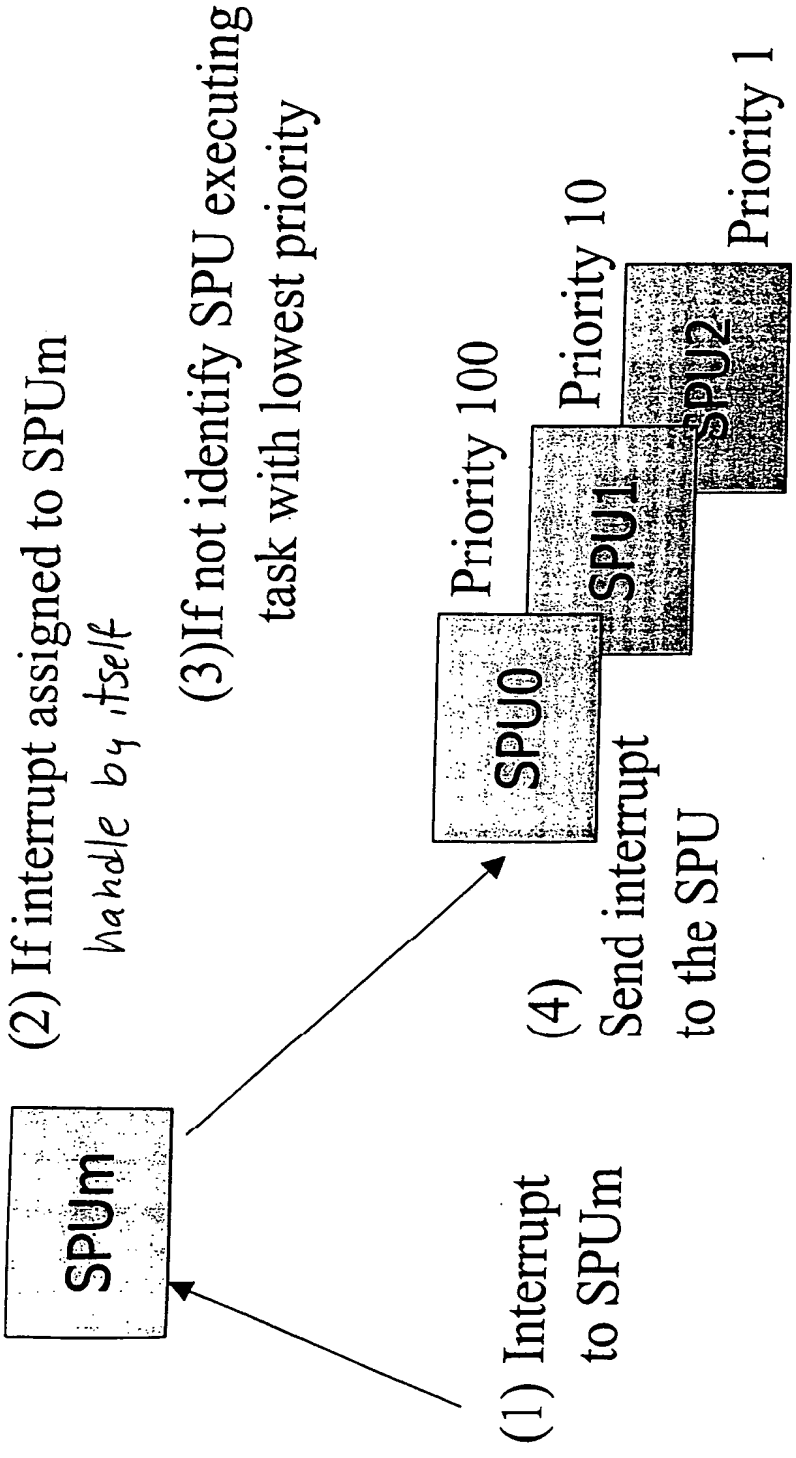

SPUm
(1) Interrupt to SPUm
(2) If interrupt assigned to SPUm handle by itself
(3)If not identify SPU executing task with lowest priority
(4) Send interrupt to the SPU
SPU0
Priority 100
SPU1
Priority 10
SPU2
Priority 1

FIG. 29

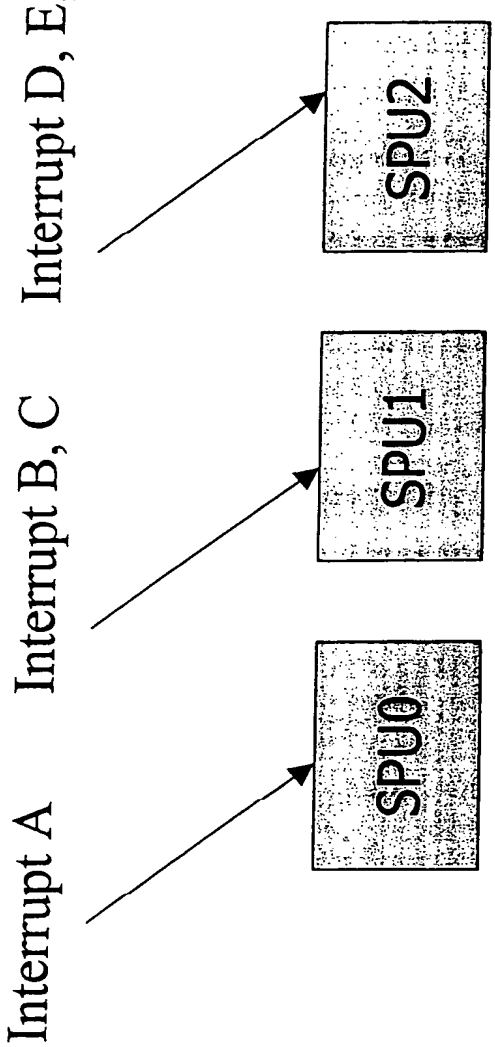


FIG. 30

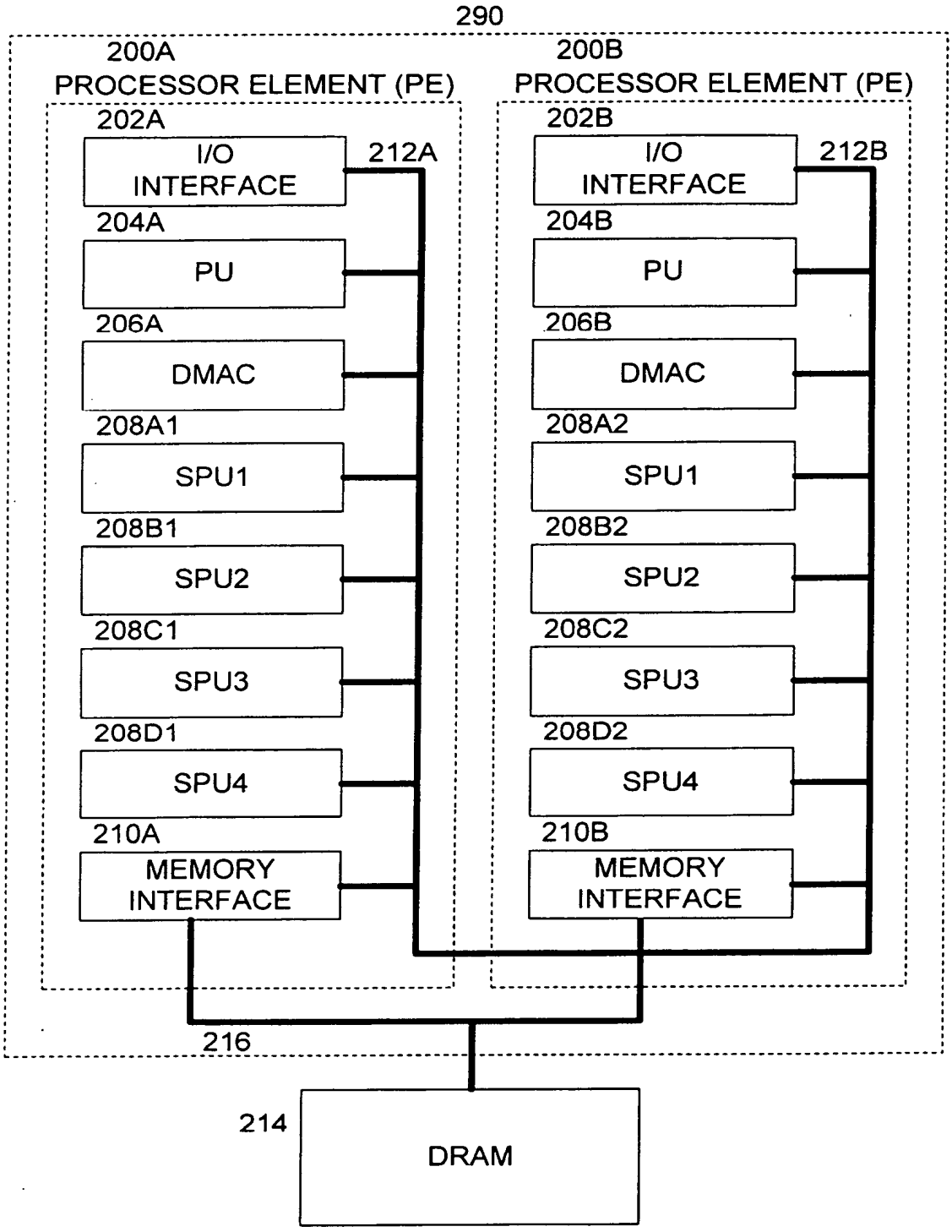

290
200A
PROCESSOR ELEMENT (PE)
200B
PROCESSOR ELEMENT (PE)
202A
I/O INTERFACE
212A
202B
I/O INTERFACE
212B
204A
PU
204B
PU
206A
DMAC
206B
DMAC
208A1
SPU1
208A2
SPU1
208B1
SPU2
208B2
SPU2
208C1
SPU3
208C2
SPU3
208D1
SPU4
208D2
SPU4
210A
MEMORY INTERFACE
210B
MEMORY INTERFACE
216
214
DRAM

FIG. 31

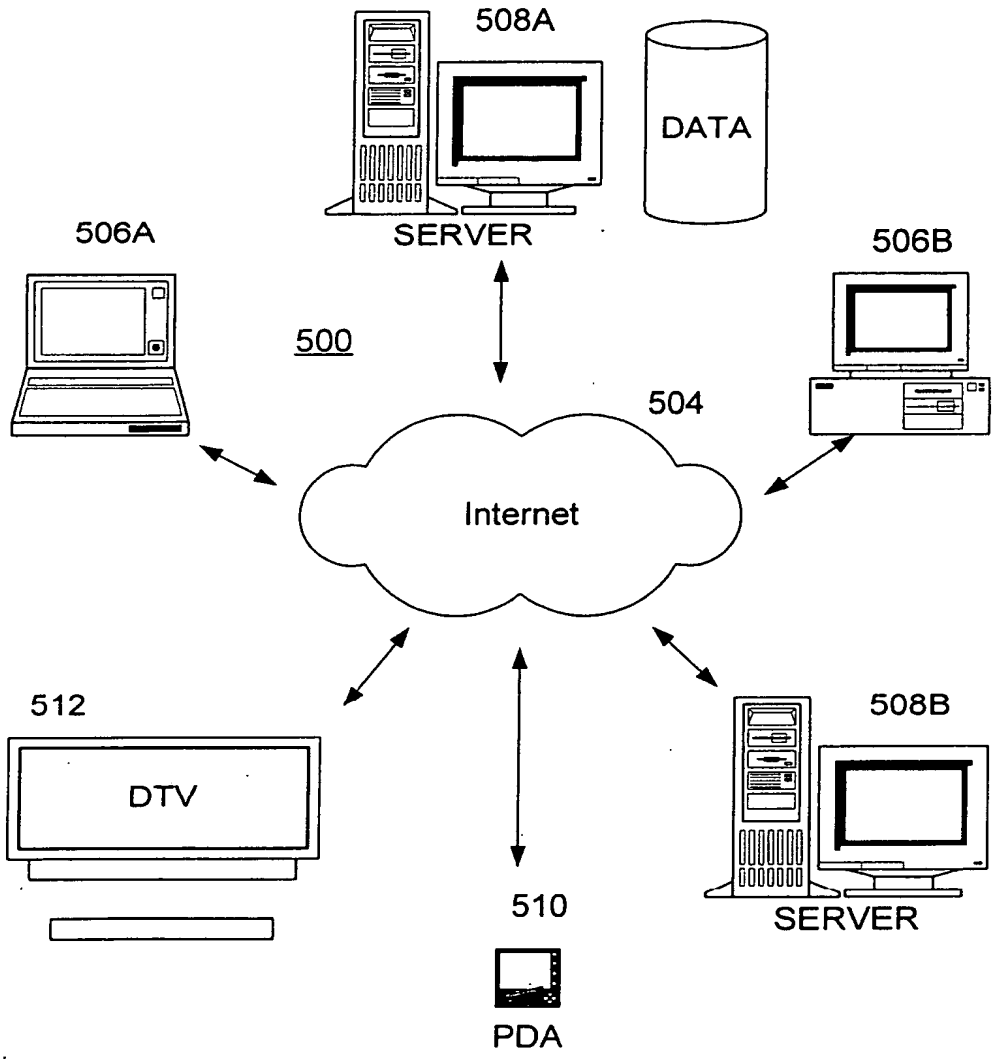
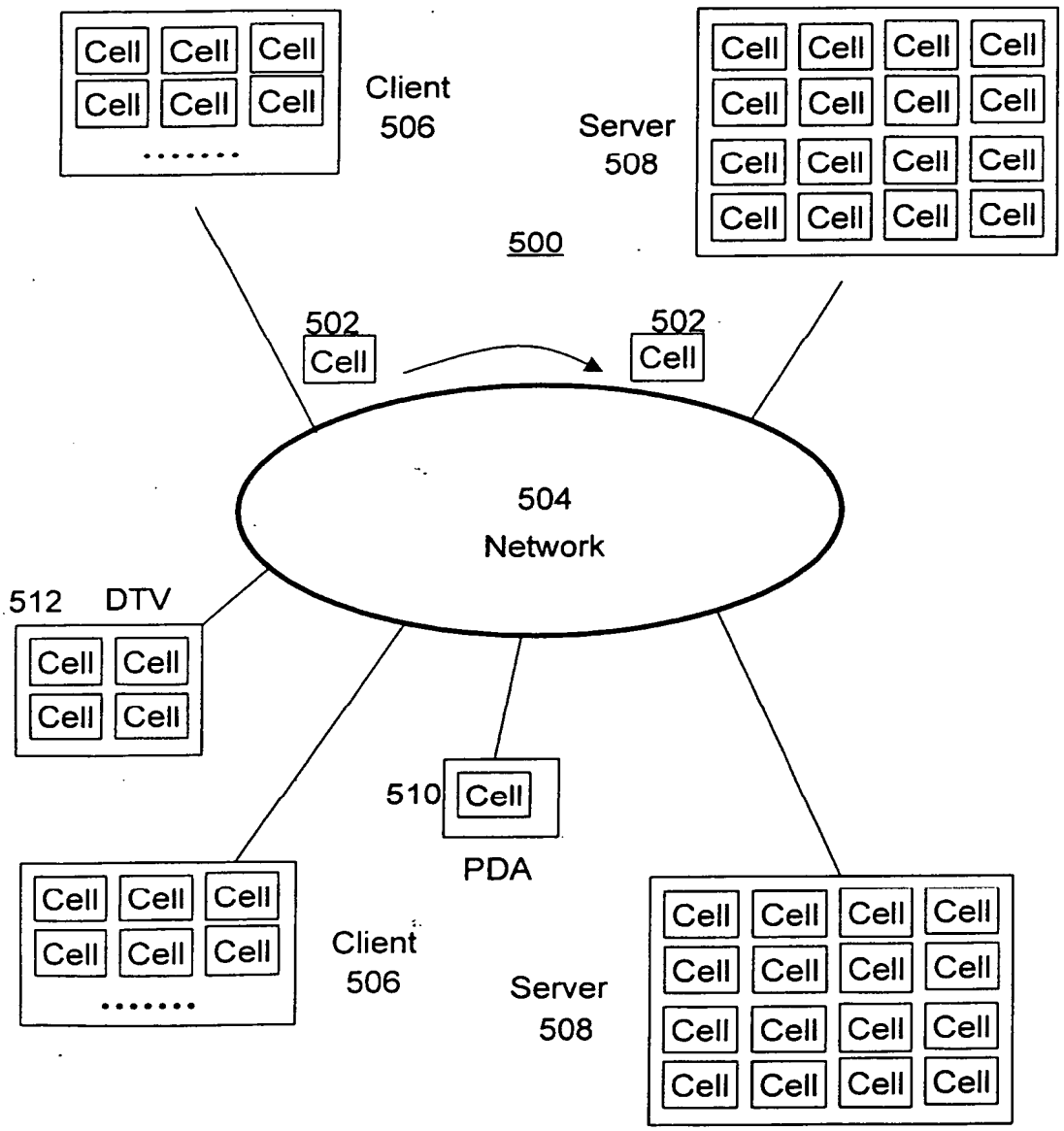

508A
DATA
SERVER
506A
506B
500
504
Internet
512
DTV
508B
SERVER
510
PDA

FIG. 32


Cell
Client
506
Server
508
500
502
Cell
504
Network
512 DTV
510
PDA